Contents

Request Timeouts

3 minute read ✔ page test

This task shows you how to setup request timeouts in Envoy using Istio.

Before you begin

- Setup Istio by following the instructions in the Installation guide.
- Deploy the Bookinfo sample application including the default destination rules.
- Initialize the application version routing by running the following command:

```
$ kubectl apply -f @samples/bookinfo/networking/virtual-service-all-v1
.yaml@
```

Request timeouts

A timeout for HTTP requests can be specified using the *timeout* field of the route rule. By default, the request timeout is disabled, but in this task you override the `reviews` service timeout to 1 second. To see its effect, however, you also introduce an artificial 2 second delay in calls to the `ratings` service.

1. Route requests to v2 of the `reviews` service, i.e., a version

that calls the `ratings` service:

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v2
EOF
```

2. Add a 2 second delay to calls to the `ratings` service:

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: ratings
spec:
  hosts:
  - ratings
  http:
  - fault:
      delay:
        percent: 100
        fixedDelay: 2s
    route:
    - destination:
        host: ratings
        subset: v1
EOF
```

3. Open the Bookinfo URL `http://$GATEWAY_URL/productpage` in your browser.

You should see the Bookinfo application working normally (with ratings stars displayed), but there is a 2 second delay whenever you refresh the page.

4. Now add a half second request timeout for calls to the `reviews` service:

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v2
    timeout: 0.5s
EOF
```

5. Refresh the Bookinfo web page.

   You should now see that it returns in about 1 second, instead of 2, and the reviews are unavailable.

The reason that the response takes 1 second, even though the timeout is configured at half a second, is because there is a hard-coded retry in the `productpage` service, so it calls the timing out `reviews` service twice before returning.

Understanding what happened

In this task, you used Istio to set the request timeout for calls to the `reviews` microservice to half a second. By default the

request timeout is disabled. Since the `reviews` service subsequently calls the `ratings` service when handling requests, you used Istio to inject a 2 second delay in calls to `ratings` to cause the `reviews` service to take longer than half a second to complete and consequently you could see the timeout in action.

You observed that instead of displaying reviews, the Bookinfo product page (which calls the `reviews` service to populate the page) displayed the message: Sorry, product reviews are currently unavailable for this book. This was the result of it receiving the timeout error from the `reviews` service.

If you examine the fault injection task, you'll find out that the `productpage` microservice also has its own application-level

timeout (3 seconds) for calls to the `reviews` microservice. Notice that in this task you used an Istio route rule to set the timeout to half a second. Had you instead set the timeout to something greater than 3 seconds (such as 4 seconds) the timeout would have had no effect since the more restrictive of the two takes precedence. More details can be found `here`.

One more thing to note about timeouts in Istio is that in addition to overriding them in route rules, as you did in this task, they can also be overridden on a per-request basis if the application adds an `x-envoy-upstream-rq-timeout-ms` header on outbound requests. In the header, the timeout is specified in milliseconds instead of seconds.

Cleanup

- Remove the application routing rules:

```
$ kubectl delete -f @samples/bookinfo/networking/virtual-service-all-v
1.yaml@
```

- If you are not planning to explore any follow-on tasks, see the Bookinfo cleanup instructions to shutdown the application.